Verified global optimization for estimating the parameters of nonlinear models

Michel Kieffer, Mihaly Csaba Markot, Hermann Schichl, Eric Walter

Chapter 1
Verified global optimization for estimating the parameters of nonlinear models

Michel Kieffer (✉), Mihály Csaba Markót, Hermann Schichl, and Eric Walter

Abstract Nonlinear parameter estimation is usually achieved via the minimization of some possibly non-convex cost function. Interval analysis allows one to derive algorithms for the guaranteed characterization of the set of all global minimizers of such a cost function when an explicit expression for the output of the model is available or when this output is obtained via the numerical solution of a set of ordinary differential equations. However, cost functions involved in parameter estimation are usually challenging for interval techniques, if only because of multi-occurrences of the parameters in the formal expression of the cost. This paper addresses parameter estimation via the verified global optimization of quadratic cost functions. It introduces tools for the minimization of generic cost functions. When an explicit expression of the output of the parametric model is available, significant improvements may be obtained by a new box exclusion test and by careful manipulations of the quadratic cost function. When the model is described by ODEs, some of the techniques available in the previous case may still be employed, provided that sensitivity functions of the model output with respect to the parameters are available.

1.1 Introduction

Estimating the parameters of models from experimental data often involves the optimization of possibly non-convex cost functions. Let $\mathbf{y}(t_i)$ be the vector of all mea-

Michel Kieffer and Eric Walter
Laboratoire des Signaux et Systèmes - CNRS - SUPELEC - Univ Paris-Sud, 3 rue Joliot-Curie, F-91192 Gif-sur-Yvette cedex e-mail: kieffer@lss.supelec.fr, walter@lss.supelec.fr

Michel Kieffer
on leave at LTCI - CNRS - Télécom ParisTech, 46 rue Barrault, F-75013 Paris

Mihály Csaba and Markót, Hermann Schichl
Fakultät für Mathematik, Universität Wien, Nordbergstr. 15, A-1090 Wien, Austria e-mail: Mihaly.Markot@univie.ac.at, Hermann.Schichl@univie.ac.at

surements collected on the system to be modeled at some time instant t_i, $i = 1, \ldots, N$. The model output $\mathbf{y}_{\mathrm{m}}(\mathbf{x}, t_i)$ at some instant t_i may consist of an explicit expression involving the vector $\mathbf{x}$ of parameters to be estimated, or it may require the solution of sets of ordinary differential equations (ODEs) containing $\mathbf{x}$ such as

$$\frac{\mathrm{d}\mathbf{z}}{\mathrm{d}t} = \mathbf{g}(\mathbf{z}, \mathbf{x}, t)\,,\ \mathbf{z}(t_0) = \mathbf{z}_0, \tag{1.1}$$

where $\mathbf{z}$ is some state vector with initial value $\mathbf{z}_0$ at t_0 and with

$$\mathbf{y}_{\mathrm{m}}(\mathbf{x}, t) = \mathbf{h}(\mathbf{z}, \mathbf{x}, t)\,.$$

A standard procedure for estimating $\mathbf{x}$ (see, *e.g.*, [9,38] and the references therein) is via the minimization of a cost function $f(\mathbf{x})$, which may be deduced from probabilistic assumptions on the noise affecting the measurements and on the parameters. Often, this cost function is quadratic, for instance

$$f(\mathbf{x}) = (\mathbf{y}_{\mathrm{m}}(\mathbf{x}) - \mathbf{y})^{\mathrm{T}} (\mathbf{y}_{\mathrm{m}}(\mathbf{x}) - \mathbf{y})\,, \tag{1.2}$$

where

$$\mathbf{y}_{\mathrm{m}}^{\mathrm{T}}(\mathbf{x}) = \left(\mathbf{y}_{\mathrm{m}}^{\mathrm{T}}(\mathbf{x}, t_1), \ldots, \mathbf{y}_{\mathrm{m}}^{\mathrm{T}}(\mathbf{x}, t_N)\right) \tag{1.3}$$

and

$$\mathbf{y}^{\mathrm{T}} = \left(\mathbf{y}^{\mathrm{T}}(t_1), \ldots, \mathbf{y}^{\mathrm{T}}(t_N)\right). \tag{1.4}$$

When $\mathbf{y}_{\mathrm{m}}(\mathbf{x})$ is linear in $\mathbf{x}$, the minimization of a quadratic $f(\mathbf{x})$ is, up to numerical stability issues, a trivial matter. Unfortunately, many models are actually nonlinear in $\mathbf{x}$, *e.g.*, knowledge-based models such as those encountered in physics, chemistry, or biology. As a consequence, $f(\mathbf{x})$ may admit in some cases several global minimizers that are all equally valid estimates. The usual local methods (such as those based on Gauss-Newton or conjugate gradient algorithms) then converge at best to a local minimizer of the cost function. Global optimization methods based on random search (for instance simulated annealing or genetic algorithms) cannot guarantee to locate all global minimizers in finite time.

Guaranteed optimization algorithms based on interval analysis [5, 11, 22, 23, 39], on the other hand, are able to derive *proven* statements about the global minimum of the cost function and associated set of global minimizers. However, cost functions involved in parameter estimation are usually challenging for interval techniques, due, *e.g.*, to multi-occurrences of the vector of parameters in the expression of the cost function [11, 22]. Getting tight enclosures of cost functions over large boxes is then very difficult. This, combined with the curse of dimensionality, restrains the dimension of problems, which may be addressed using such guaranteed techniques.

The aim of this chapter is to provide some results which may help improving the efficiency of global optimization using interval techniques, especially in the case of cost functions used in parameter estimation. Tools for the guaranteed minimization of generic cost functions are first recalled in Section 1.2. Section 1.3 then focuses on techniques that significantly improve global optimization algorithms, such as

constraint propagation, a new box exclusion test, and symbolic manipulations of the cost function. Such manipulations are possible when an explicit expression of the output of the parametric model is available. When the model is described by ODEs, some of the techniques introduced in Sections 1.2 and 1.3 may still be employed, provided that sensitivity functions of the model output with respect to the parameters are available, see Section 1.3.4. Improvements provided by the tools presented to the problem of parameter estimation via verified global optimization of quadratic cost functions are illustrated on a simple compartmental model in Section 1.4.

1.2 Basics of guaranteed optimization

This section recalls some well-known methods for guaranteed optimization that are relevant for nonlinear parameter estimation.

1.2.1 Problem formulation

Consider the generic bound-constrained optimization problem

$$\begin{aligned} &\min\ f(\mathbf{x}), \\ &\text{s.t.}\ \ \mathbf{x} \in [\mathbf{x}]_0, \end{aligned} \tag{1.5}$$

where $[\mathbf{x}]_0 \in \mathbb{IR}^n$ is some search box, and the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is at least twice continuously differentiable on $[\mathbf{x}]_0$. The problem of parameter estimation via minimization of some cost function may be written as (1.5), provided that a (possibly very large) initial search box $[\mathbf{x}]_0$ has been chosen.

As already mentioned in the introduction, the aim of *deterministic* global optimization is to find *rigorous* interval enclosures to *all* global minimizers and to the global minimum f^*. The most widely used scheme of interval-based global optimization methods is the branch–and–bound (B&B) technique introduced by [12, 14] for discrete problems and for continuous problems by [18, 35]. There have been numerous improvements, see [22] for a recent survey.

1.2.2 Why is global optimization for parameter estimation difficult?

Assume, for the sake of simplicity, that $y(t_i)$ is scalar, so the objective function (1.2) may be rewritten as

$$f(\mathbf{x}) = \sum_{i=1}^{N} (y_{\mathrm{m}}(\mathbf{x}, t_i) - y_i)^2. \tag{1.6}$$

The cost function (1.6) consists of N squared differences between $y_{\mathrm{m}}(\mathbf{x},t_i)$ and y_i. Each of these squares may involve several occurrences of the parameter vector $\mathbf{x}$, leading to at least N occurrences of $\mathbf{x}$ in the expression of (1.6). Getting accurate inclusion functions for f thus may be particularly challenging. Moreover, the function evaluation near the minimizers is often dominated by cancellation since the y_is and the $y_{\mathrm{m}}(\mathbf{x},t_i)$s are often magnitudes higher than their difference. This often causes severe overestimation in the interval evaluations, which slows down branch-and-bound methods and increases the cluster effect.

1.2.3 Interval branch–and–bound methods

In general, interval B&B involves the following main iteration loop (the terminology *working list* refers to the subboxes waiting for further processing, *i.e.*, those located at *open* leaves of the B&B tree):

1. Step 1: select a subbox $[\mathbf{x}] \subseteq [\mathbf{x}]_0$ from the working list;
2. Step 2: split $[\mathbf{x}]$ into subboxes $[\mathbf{x}]_i$, $i = 1,\ldots,k$;
3. Step 3: for each i run acceleration tests to eliminate $[\mathbf{x}]_i$ or parts of it that cannot contain a global minimizer;
4. Step 4: if the stopping criterion holds for the remaining part of $[\mathbf{x}]_i$, then store it in the result list R, else store the remaining part of $[\mathbf{x}]_i$ in the working list;
5. Step 5: update the best known upper bound $\tilde{f}$ of the global minimum using information acquired from $[\mathbf{x}]_i$.

Initially the working list contains only $[\mathbf{x}]_0$. The main loop is executed until the working list becomes empty. At the end of the algorithm it is ensured that enclosures of all global minimizers are in R, and the global minimum is in the interval $[\min_{[\mathbf{x}]\in R} \inf(f([\mathbf{x}])), \tilde{f}]$.

For all steps of the B&B algorithm, there exist a number of tools. Here we will focus on describing those tools that were most successful in solving nonlinear least squares problems using the `coco_gop_ex` solver (see Section 1.4.1.4). However, the methods presented in what follows are quite general and can be applied to solving all kinds of global optimization problems, see [17].

1.2.3.1 Operations on the working list

The subbox to be subdivided is selected as the one with the smallest lower bound of interval enclosure for the cost (Moore-Skelboe rule). The boxes are split into two subboxes in a direction determined by a first order merit function given by Csendes and Ratz (rule 'B' in [24]). The stopping criterion used in Step 4 is $\mathrm{diam}(f([\mathbf{y}])) < \varepsilon$ for a pre-specified tolerance value ε.

1.2.3.2 Tools to update $\tilde{f}$

For every subbox we compute $\sup(f([\mathbf{c}]))$ for the interval enclosure of the center $\mathbf{c}$ to update $\tilde{f}$. Furthermore, if $\sup(f([\mathbf{c}])) < \tilde{f}$, we run a local search from $\mathbf{c}$.

1.2.3.3 Tools to prune or erase $[\mathbf{x}]$

Most of the effort for solving a global optimization problem is spent in this phase of the solver. The effectivity of the implemented pruning or reduction techniques for subboxes is essential for the efficiency of the B&B solver.

Bound (suboptimality) test

If $\inf(V_f([\mathbf{x}])) > \tilde{f}$, then the box $[\mathbf{x}]$ cannot contain a global minimizer and may be discarded. For this test an enclosure $f([\mathbf{x}])$ of the range $V_f([\mathbf{x}])$ of the objective function on $[\mathbf{x}]$ is computed by *interval evaluation*. There are several methods for computing enclosures for the values taken by a function over a box, naive interval arithmetic being the one that requires the least effort. Naive interval arithmetic usually overestimates the range, however, and the overestimation is proportional to the radius of $[\mathbf{x}]$. This is a problem, since it often makes it impossible to eliminate boxes that are close to a global minimizer without further splitting them. Therefore, estimation methods with higher order approximation properties, *i.e.*, overestimation being $O(\mathrm{rad}([\mathbf{x}]))^p$ for $p > 1$, are needed to remove boxes close to a global minimizer. Centered forms and higher-order centered forms provide such estimates. They can be based on interval gradients or higher interval derivatives or on slopes of first or higher order (see [21, 29, 33]). Typical centered forms used to get tigher enclosures are

$$
\begin{aligned}
V_f([\mathbf{x}]) &\subseteq f(\mathbf{z}) + \nabla f([\mathbf{x}])^T([\mathbf{x}]-\mathbf{z}) \\
V_f([\mathbf{x}]) &\subseteq f(\mathbf{z}) + (\nabla f(\mathbf{z})^T + \tfrac{1}{2}([\mathbf{x}]-\mathbf{z})^T \nabla^2 f([\mathbf{x}]))([\mathbf{x}]-\mathbf{z}), \\
V_f([\mathbf{x}]) &\subseteq f(\mathbf{z}) + \big(\nabla f(\mathbf{z})^T + \tfrac{1}{2}([\mathbf{x}]-\mathbf{z})^T \big(\nabla^2 f(\mathbf{z}) \\
&\qquad + \tfrac{1}{3}\sum_i \nabla^3_{i::} f([\mathbf{x}])([\mathbf{x}]_i - z_i)\big)\big)([\mathbf{x}]-\mathbf{z}),
\end{aligned}
$$

where $\mathbf{z}$ is usually chosen as the center of the box $[\mathbf{x}]$, and the notation : indicates that all possible values of the index should be considered. The first has quadratic, the next cubic, and the last one quartic approximation property.

Here, the naive interval arithmetic and the first two centered forms above have been considered.

Monotonicity test

If $0 \notin V_{\nabla_i f}([\mathbf{x}])$ for some i, then $[\mathbf{x}]$ cannot contain a global minimizer in its interior. The range of the gradient over some $[\mathbf{x}]$ can also be enclosed using various methods. It can be computed by forward and backward algorithmic differentiation [28, 33], the forward evaluation giving better enclosures but taking an effort of $O(n)$ function evaluations, while the backward method produces slightly worse enclosures but requires an effort of only about two function evaluations. For both approaches the overestimation is $O(\mathrm{rad}([\mathbf{x}]))$. Centered forms can be used to get higher-order approximation properties for the gradient as well, thus increasing the effectiveness of the monotonicity test for small boxes close to a critical point. Typical centered forms are

$$V_{\nabla f}([\mathbf{x}]) \subseteq \nabla f(\mathbf{z}) + \nabla^2 f([\mathbf{x}])([\mathbf{x}] - \mathbf{z}),$$
$$V_{\nabla f}([\mathbf{x}]) \subseteq \nabla f(\mathbf{z}) + \big(\nabla^2 f(\mathbf{z}) + \tfrac{1}{2}\sum_i \nabla^3_{i::} f([\mathbf{x}])([\mathbf{x}]_i - z_i)\big)\big)([\mathbf{x}] - \mathbf{z}),$$

where again $\mathbf{z}$ usually is chosen as the center of the box $[\mathbf{x}]$. The first centered form has quadratic and the second cubic approximation property. Here, the first centered form update of the interval gradient has only been used, whenever an interval Hessian has been computed.

Interval Newton test

A Gauss-Seidel iteration is used to solve the interval system

$$\nabla^2 f([\mathbf{x}]) \cdot ([\mathbf{x}] - \mathbf{c}) + \nabla f(\mathbf{c}) = 0, \tag{1.7}$$

with $\mathbf{c} \in [\mathbf{x}]$, to verify the uniqueness or non-existence of a stationary point in $[\mathbf{x}]$ [21]. The interval Newton test can shrink $[\mathbf{x}]$ or return a set of subboxes of $[\mathbf{x}]$ that needs to be considered further in place of $[\mathbf{x}]$.

Constraint propagation (CP)

Since every global minimizer $\mathbf{x}^*$ of (1.5) has to satisfy $f(\mathbf{x}^*) \leq \tilde{f}$, the additional constraint $f(\mathbf{x}) \leq \tilde{f}$ may be introduced. We attempt to reduce $[\mathbf{x}]$ by propagating this information back to the variables. An especially efficient method for constraint propagation is the PAID propagator [36], see Section 1.3.1. It is based on coordinating forward evaluation and backward propagation steps to reduce the bounds on all variables and intermediate nodes as much as possible.

For the least squares situation (1.6), the constraint propagator will, *e.g.*, automatically take into account the N additional constraints

$$y_{\mathrm{m}}(\mathbf{x},t_i) - y_i \in \left[-\sqrt{\tilde{f}}, \sqrt{\tilde{f}}\right], \tag{1.8}$$

which may be deduced from the fact that if $f(\mathbf{x}) \leq \tilde{f}$, then each term of the sum in (1.6) has to satisfy $(y_{\mathrm{m}}(\mathbf{x},t_i) - y_i)^2 \leq \tilde{f}$, which may be rewritten as (1.8).

Exclusion/inclusion boxes

To avoid the cluster effect [10] higher-order methods are necessary. These are usually invoked right after a new approximate local minimizer $\tilde{\mathbf{x}}$ has been detected, in order to provide a pair of boxes $\tilde{\mathbf{x}} \in [\mathbf{x}]^{\mathrm{i}} \subseteq [\mathbf{x}]^{\mathrm{e}}$. In the inclusion box $[\mathbf{x}]^{\mathrm{i}}$ uniqueness of the local optimizer is proved, along with non-existence of another local optimizer in the interior of the exclusion box $[\mathbf{x}]^{\mathrm{e}}$. This box $[\mathbf{x}]^{\mathrm{e}}$ then significantly reduces the size of the result list, since boxes are pruned from the search tree, whenever they are interior to $[\mathbf{x}]^{\mathrm{e}}$. A more detailed description of this technique can be found in Section 1.3.2.

1.3 Improving the efficiency of guaranteed techniques

1.3.1 The PAID constraint propagator

For the PAID propagator the cost function needs to be represented as a directed acyclic graph of elementary operations, called the *model DAG* in what follows. The Forward-Backward Propagation on DAGs (FBPD) algorithm is used to compute and improve enclosures of the ranges of all nodes in the DAG. Let $\mathbf{N}$ be a node that has k children $\{\mathbf{C}_i\}_{i=1}^{k}$, denoting its input variables. The elementary operation represented by $\mathbf{N}$ is a function $g : D_g \to \mathbb{R}$, where $D_g \subseteq \mathbb{R}^k$ denotes the domain of definition of g. Hence, the relationship between $\mathbf{N}$ and its children can be written as $\mathbf{N} = g(\mathbf{C}_1, \ldots, \mathbf{C}_k)$. Let $[g]$ be an inclusion function of g. The *forward evaluation* at node $\mathbf{N}$ using the inclusion function $[g]$ is defined as

$$\mathtt{FE}(\mathbf{N}, [g]) \equiv \{\mathbb{D}(\mathbf{N}) := \mathbb{D}(\mathbf{N}) \cap [g]\}, \tag{1.9}$$

where $\mathbb{D}(\mathbf{M})$ denotes the currently best known enclosure for node $\mathbf{M}$. This forward evaluation computes the enclosure of the range of a node based on the enclosures of the ranges of its children (its input variables) using an inclusion function of the elementary operation representing that node.

The *backward propagation* prunes the enclosures associated with children based on the constraint range of their parent. In other words, for each child $\mathbf{C}_i$ the backward propagation evaluates the i-th projection of the relation $\mathbf{N} = g(\mathbf{C}_1, \ldots, \mathbf{C}_k)$ on the input variable represented by $\mathbf{C}_i$. We call it the i-th backward propagation at $\mathbf{N}$ and denote it by $\mathtt{BP}(\mathbf{N}, \mathbf{C}_i)$. We define the following sequence as the backward

propagation at node $\mathbf{N}$

$$\texttt{BP}(\mathbf{N}) = \{\texttt{BP}(\mathbf{N}, \mathbf{C}_i)\}_{i=1}^{k}. \tag{1.10}$$

Although the exact projection of relations is expensive in general, an evaluation of the exact projection of elementary operations can be obtained at low cost. Indeed, assume that from the relation $\mathbf{N} = g(\mathbf{C}_1, \ldots, \mathbf{C}_k)$ one can infer an equivalent relation $\mathbf{C}_i = h_i(\mathbf{N}, \{\mathbf{C}_j\}_{j=1}^{k})$ for some $i \in \{1, \ldots, k\}$, where h_i is a function from $\mathbb{R}^k$ to $\mathbb{R}$. Let $[h_i]$ be an inclusion function of h_i. The i-th backward propagation can then be obtained as follows

$$\texttt{BP}\,(\mathbf{N}, \mathbf{C}_i) \equiv \left\{ \mathbb{D}(\mathbf{C}_i) := \mathbb{D}(\mathbf{C}_i) \cap [h_i] \left(\mathbb{D}(\mathbf{N}), \{\mathbb{D}(\mathbf{C}_j)\}_{j=1}^{k} \right) \right\}. \tag{1.11}$$

The FBPD algorithm coordinates forward and backward steps through the model DAG by a proper ordering of the nodes. The overall scheme is independent of the type of enclosure chosen at each node. Most usual are interval enclosures of the range, but interval sets, affine enclosures, and centered-form enclosures, as well as combinations of them are also possible. The complete algorithm can be found in [36].

The PAID propagator can also be used for bound and monotonicity tests, especially if it is combined with the Karush-John generator which computes a DAG representation of the first order optimality conditions. This is very efficient for general nonlinear problems with equality and inequality constraints. However, in the case of nonlinear least squares problems, the efficiency is limited, and for the parameter estimation problem considered the overall solution time is actually about 30% higher if PAID is enabled for checking the first order optimality conditions.

1.3.2 Exclusion and Inclusion Boxes

Close to a global minimizer it is usually difficult to remove subboxes generated during the splitting phase. In [10], it was shown that avoiding the cluster effect requires at least second-order methods. For very flat problems, such as nonlinear least-squares problems, close to global minimizers second order information is usually not enough. Based on [32], we have developed in [31] a third-order method that computes large exclusion boxes for optimization problems. For `coco_gop_ex` we have implemented the special case for unconstrained problems, which can be applied for optima in the interior of $[\mathbf{x}]_0$.

Let $\mathbf{z}$ be an approximate local solution of (1.5) in the interior of $[\mathbf{x}]_0$. We compute the preconditioning matrix $\mathbf{C} \approx (\nabla^2 f(\mathbf{z}))^{-1}$ as the inverse of the point Hessian at $\mathbf{z} \in [\mathbf{x}]$ for some box $[\mathbf{x}] \subseteq [\mathbf{x}]_0$. Using this we compute the following estimates (using directed rounding and interval arithmetic)

$$\overline{\mathbf{b}} \geq |\mathbf{C}\nabla f(\mathbf{z})|, \qquad \mathbf{B}_0 \geq |\mathbf{C}\nabla^2 f(\mathbf{z}) - \mathbf{I}|,$$
$$\mathscr{B}_{ijk} \geq \tfrac{1}{2}\left|\sum_l C_{il}\nabla^3_{ljk} f([\mathbf{x}])\right|,$$

as tightly as feasible, where $|\cdot|$ denotes the componentwise absolute value. Choose some $\mathbf{v} \in \mathbb{R}^n$ with $\mathbf{v} > 0$ and set $\mathbf{w} := (\mathbf{I} - \mathbf{B}_0)\mathbf{v}$, $a_i := \sum_{j,k} v_j \mathscr{B}_{ijk} v_k$. If $D_j = w_j^2 - 4a_j\overline{b}_j > 0$ for all j, define

$$\lambda_j^{\mathrm{e}} := \frac{w_j + \sqrt{D_j}}{2a_j}, \quad \lambda_j^{\mathrm{i}} := \frac{\overline{b}_j}{a_j \lambda_j^{\mathrm{e}}}, \quad \lambda^{\mathrm{e}} := \min_j \lambda_j^{\mathrm{e}}, \quad \lambda^{\mathrm{i}} := \max_j \lambda_j^{\mathrm{i}}.$$

Theorem 1. *Let all estimates above be satisfied for the box* $[\mathbf{x}]$. *If now* $D_j > 0$ *for all* j *and* $\lambda^e > \lambda^i$ *then there exists a unique critical point* $\mathbf{x}^*$ *for (1.5) in the inclusion region* $[\mathbf{x}]^i := [\mathbf{z} - \lambda^i\mathbf{v}, \mathbf{z} + \lambda^i\mathbf{v}] \cap [\mathbf{x}]$. *This is the only critical point of (1.5) in the interior of the exclusion region* $[\mathbf{x}]^e := [\mathbf{z} - \lambda^e\mathbf{v}, \mathbf{z} + \lambda^e\mathbf{v}] \cap [\mathbf{x}]$.

Proof. Only a general idea of the proof is provided here, see [31] for more details. The result follows from [32, Theorem 4.3] if we set $\mathbf{f} = \nabla f(\mathbf{x})$.

In [8] and [34] it was shown that existence and uniqueness of a zero of a C^1-function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ in the box $[\mathbf{x}]$ may be proved using the Krawczyk operator

$$K_0([\mathbf{x}], \mathbf{z}) := \mathbf{z} - \mathbf{C}\mathbf{f}(\mathbf{z}) - (\mathbf{C}\nabla\mathbf{f}([\mathbf{x}]) - \mathbf{I})([\mathbf{x}] - \mathbf{z}),$$

where $\mathbf{I}$ is the identify matrix and $\mathbf{C}$ is some arbitrary matrix. More precisely, if $K_0([\mathbf{x}], \mathbf{z}) \subseteq [\mathbf{x}]$ then $[\mathbf{x}]$ contains a zero of $\mathbf{f}$. If $K_0([\mathbf{x}], \mathbf{z}) \subseteq \mathrm{int}([\mathbf{x}])$ then there is a *unique* zero in $[\mathbf{x}]$.

Instead of K_0 we can consider the second-order Krawczyk-type operator [31]

$$K([\mathbf{x}], \mathbf{z}) := \mathbf{z} - \mathbf{C}\mathbf{f} - \left(\mathbf{C}\nabla\mathbf{f}(\mathbf{z}) - \mathbf{I} + \sum_{k=1}^{n} ([\mathbf{x}]_k - \mathbf{z}_k)^{\mathrm{T}} \nabla^2 \mathbf{f}([\mathbf{x}])_{::k}\right)([\mathbf{x}] - \mathbf{z}). \quad (1.12)$$

Then K has the same properties as K_0 with regard to proving existence and uniqueness of zeros of $\mathbf{f}$.

A critical point $\mathbf{x}^*$ for f in (1.5) satisfies $\nabla f(\mathbf{x}^*) = 0$, so we set $\mathbf{f} = \nabla f(\mathbf{x})$. Then we prove that for every $\mathbf{z} \in [\mathbf{x}]$ and every critical point $x^* \in [\mathbf{x}]$ the deviation $\mathbf{s} := |\mathbf{x}^* - \mathbf{z}|$ satisfies

$$0 \leq \mathbf{s} \leq \left(\mathbf{B}_0 + \sum \mathbf{s}_k \mathscr{B}_{::k}(x)\right)\mathbf{s} + \overline{\mathbf{b}}.$$

This is then used to prove that for $\mathbf{u} \in \mathbb{R}^n$, $\mathbf{u} > \mathbf{0}$, with

$$\left(\mathbf{B}_0 + \sum \mathbf{u}_k \mathscr{B}_{::k}\right)\mathbf{u} + \overline{\mathbf{b}} < \mathbf{u} \quad (1.13)$$

the set $\{\mathbf{x} \in [\mathbf{x}] \mid |\mathbf{x} - \mathbf{z}| \leq \mathbf{u}\}$ contains a unique critical point of $\mathbf{f}$. Finally, to find such an $\mathbf{u}$ we choose an arbitrary $\mathbf{v} \in \mathbb{R}^n$, $\mathbf{v} > \mathbf{0}$ and set $\mathbf{u} := \lambda\mathbf{v}$. Equation (1.13)

then leads to a quadratic equation in λ for every component of $\mathbf{u}$, which in turn proves the theorem. □

The Krawczyk-type operator (1.12) takes for $\mathbf{f} = \nabla f$ the form

$$K([\mathbf{y}],\mathbf{z}) = \mathbf{z} - \mathbf{C}\nabla f(\mathbf{z}) - (\mathbf{C}\nabla^2 f(\mathbf{z}) - \mathbf{I} + \sum_{k=1}^{n} ([\mathbf{y}]_k - \mathbf{z}_k)^T \nabla^3_{::k} f([\mathbf{y}]))([\mathbf{y}] - \mathbf{z}).$$

This operator can be used after computing the inclusion/exclusion box pair in the third-order iteration $[\mathbf{y}]^{k+1} := K([\mathbf{y}]^k, \mathrm{mid}([\mathbf{y}]^k)) \cap [\mathbf{y}]^k$ with $[\mathbf{y}]^0 = [\mathbf{x}]^i$ to further shrink the size of the inclusion box. Usually, this contracts the inclusion box in a few iterations to the limit accuracy of floating point computations. Since third-order information is used in the iteration, on the final inclusion box the enclosure of the global minimum can be computed by the third order centered form

$$V_f([\mathbf{x}]) \subseteq f(\mathbf{z}) + \big(\nabla f(\mathbf{z})^T + \tfrac{1}{2}([\mathbf{x}] - \mathbf{z})^T \big(\nabla^2 f(\mathbf{z}) + \tfrac{1}{3} \sum_i \nabla^3_{i::} f([\mathbf{x}])([\mathbf{x}]_i - z_i)\big)\big)([\mathbf{x}] - \mathbf{z}).$$

1.3.3 Methods requiring an explicit expression for the cost function

Overestimation is a serious problem when solving nonlinear least-squares problems. In many cases the interval range enclosures overestimate the true range by several orders of magnitude due to the structure of the functions $y_\mathrm{m}(\mathbf{x}, t_i)$ in (1.6) and the severe numerical cancellation in evaluating $y_\mathrm{m}(\mathbf{x}, t_i) - y_i$ if $\mathbf{x}$ is close to a global minimizer.

Things get even more difficult if the evaluation of y_m itself is already hampered by numerical cancellation. This usually is the case if y_m is a linear combination of exponentials, as when the ODE (1.1) is linear in the state $\mathbf{z}$.

To increase the efficiency of the solution process, therefore, it is necessary to find an expression for y_m that causes as little cancellation as possible. If it, *e.g.*, can be tweaked a little bit by factoring out such that it ends up as a product of univariate functions (even if they are fairly complicated and depend on more complicated parameters) then numerical cancellation will be significantly smaller, and the interval evaluations will produce less overestimation.

In view of this effect, and partially motivated by the present study, the COCONUT environment provides special tools for the optimal interval evaluations, *i.e.* evaluations with no overestimation, of one-dimensional functions and their higher-order derivatives, see [1, 27] and Section 1.4.1.4. User defined one-dimensional functions are represented as individual nodes in the model DAGs (see Section 1.3.1). If the necessary analytic properties of the functions are supplied (domain of definition, explicit derivatives up to the specified order, limit points and values, extremal points and the associated extrema, inflexion points, poles, etc.), then optimal interval evaluations can be obtained. Furthermore, inverse function evaluations on such

one-dimensional nodes (*i.e.*, the evaluation of an enclosure of all $\mathbf{x}$ values for which $f(\mathbf{x}) \in [r]$ holds for a fixed $[r]$) can also be performed using a one-dimensional root finding method. Inverse function evaluations are key ingredients needed by the constraint propagation method in Section 1.3.1.

Alternatively, the COCONUT system can autodetect complex univariate functions using a symbolic analysis on the DAG and can compute the required information by automatic curve tracing and algorithmic differentiation [30]. However, the enclosures computed by automatic curve tracing are only approximately fully optimal. They are computed by univariate validated root finding of derivatives and evaluation of the functions on the enclosures of their zeros. The enclosures of the optima produced by that method are usually a few factors of the machine epsilon wide.

An example of proper reformulation of a model is provided in Section 1.4.1.

1.3.4 Without explicit expression for the cost function

Apart from the evaluation of the set of values taken by the cost f over some box $[\mathbf{x}]$, the algorithms presented in Sections 1.2.3, 1.3.1, and 1.3.2 require the evaluation of the range of derivatives of the cost up to the third order with respect to the parameters. This section shows the way these quantities may be obtained when no explicit expression of cost function is available for the case of models described by ODEs such as (1.1).

1.3.4.1 Getting derivatives of the cost function

Assume, for the sake of simplicity that the cost function is given by (1.6). In this case, its gradient is

$$\nabla f(\mathbf{x}) = 2\sum_{i=1}^{N} \left(y_{\mathrm{m}}(\mathbf{x},t_i) - y_i\right) \frac{\partial y_{\mathrm{m}}(\mathbf{x},t_i)}{\partial \mathbf{x}}, \tag{1.14}$$

and its Hessian matrix is

$$\nabla^2 f(\mathbf{x}) = 2\sum_{i=1}^{N} \left(\frac{\partial^2 y_{\mathrm{m}}(\mathbf{x},t_i)}{\partial \mathbf{x}} \partial \mathbf{x}^{\mathrm{T}} + \left(y_{\mathrm{m}}(\mathbf{x},t_i) - y_i\right) \frac{\partial y_{\mathrm{m}}(\mathbf{x},t_i)}{\partial \mathbf{x}} \frac{\partial y_{\mathrm{m}}(\mathbf{x},t_i)}{\partial \mathbf{x}^{\mathrm{T}}} \right). \tag{1.15}$$

The gradient thus can be computed via the evaluation of the first-order sensitivity function of the output of the model with respect to the parameters

$$\mathbf{s}_y(\mathbf{x},t_i) = \frac{\partial y_{\mathrm{m}}(\mathbf{x},t_i)}{\partial \mathbf{x}}, \tag{1.16}$$

while the Hessian matrix requires also the evaluation of the second-order sensitivity function

$$\mathbf{s}_y^2(\mathbf{x},t_i) = \frac{\partial^2 y_{\mathrm{m}}(\mathbf{x},t_i)}{\partial \mathbf{x} \partial \mathbf{x}^{\mathrm{T}}}. \tag{1.17}$$

Assume that the model is described by (1.1) with an output at time t_i given by $y_{\mathrm{m}}(\mathbf{x},t_i) = h(\mathbf{z},\mathbf{x},t_i)$, then

$$\frac{\partial y_{\mathrm{m}}(\mathbf{x},t_i)}{\partial \mathbf{x}} = \frac{\partial \mathbf{z}^{\mathrm{T}}(\mathbf{x},t_i)}{\partial \mathbf{x}} \frac{\partial h(\mathbf{z},\mathbf{x},t_i)}{\partial \mathbf{z}} + \frac{\partial h(\mathbf{z},\mathbf{x},t_i)}{\partial \mathbf{x}}, \tag{1.18}$$

where the main difficulty comes from the evaluation of the first-order sensitivity of the state vector $\mathbf{z}$ with respect to $\mathbf{x}$

$$\mathbf{S}_z(\mathbf{x},t_i) = \frac{\partial \mathbf{z}^{\mathrm{T}}(\mathbf{x},t_i)}{\partial \mathbf{x}}. \tag{1.19}$$

Similarly, the evaluation of the Hessian matrix requires first and second-order sensitivity functions of $\mathbf{z}$ with respect to $\mathbf{x}$.

1.3.4.2 Sensitivity functions

When the model is described by ODEs such as (1.1), the sensitivity functions of $\mathbf{z}$ with respect to $\mathbf{x}$ are obtained easily by evaluating the partial derivatives of (1.1) with respect to $\mathbf{x}$, and inverting the order of derivation to get

$$\frac{\mathrm{d}\mathbf{S}_z(\mathbf{x},t)}{\mathrm{d}t} = \frac{\partial \mathbf{g}^{\mathrm{T}}(\mathbf{z},\mathbf{x},t)}{\partial \mathbf{x}} \tag{1.20}$$

and

$$\mathbf{S}_z(\mathbf{x},t_0) = \frac{\partial \mathbf{z}_0^{\mathrm{T}}(\mathbf{x})}{\partial \mathbf{x}}. \tag{1.21}$$

Obtaining the first-order sensitivity functions of $\mathbf{z}$ with respect to $\mathbf{x}$ thus requires solving a *coupled* system of ODEs consisting of (1.1) supplemented with (1.20) and (1.21)

$$\begin{cases} \frac{\mathrm{d}\mathbf{z}}{\mathrm{d}t} = \mathbf{g}(\mathbf{z},\mathbf{x},t) \\ \frac{\mathrm{d}\mathbf{S}_z(\mathbf{x},t)}{\mathrm{d}t} = \frac{\partial \mathbf{g}^{\mathrm{T}}(\mathbf{z},\mathbf{x},t)}{\partial \mathbf{x}} \end{cases} \quad \text{with} \quad \begin{cases} \mathbf{z}(t_0) = \mathbf{z}_0(\mathbf{x}) \\ \mathbf{S}_z(\mathbf{x},t_0) = \frac{\partial \mathbf{z}_0(\mathbf{x})}{\partial \mathbf{x}}. \end{cases} \tag{1.22}$$

If the dimension of the initial system of ODEs is n_z, the dimension of the coupled system of ODEs (1.22) is $n_z + n_z n_x$.

Similarly, obtaining second-order sensitivity functions of the state with respect to the parameters requires solving systems of ODEs with $n_z\left(1+n_x+n_x^2\right)$ equations. Due to the increase in complexity of the systems to be solved, higher order methods described in Section 1.3.2 are quite difficult to apply when no explicit expression of the output of the model is available.

1.3.4.3 Guaranteed numerical integration

Several approaches may be considered to solve (1.22). Classical methods for the solution of systems of ODEs, such as Runge-Kutta, are not able to provide the range of the solutions at each t_i, $i = 1,\ldots,N$ when $\mathbf{x}$ is only known to belong to some box $[\mathbf{x}]$. Guaranteed numerical integration techniques could be employed, such as `AWA` [15], `VNODE` [20], `COSY IV` [6], `VSPODE` [13], or `ValEncIA-IVP` [25,26]. The main difficulty comes from the fact that obtaining accurate enclosures for the solutions when there are uncertain parameters, as here, and uncertain initial conditions is quite difficult.

To address this issue, one may build an *extended state* $\mathbf{z}_\mathrm{e} = \left(\mathbf{z}^\mathrm{T}, \mathbf{x}^\mathrm{T}\right)^\mathrm{T}$, satisfying the following extended systems of ODEs

$$\frac{\mathrm{d}\mathbf{z}_\mathrm{e}}{\mathrm{d}t} = \begin{pmatrix} \mathbf{g}(\mathbf{z},\mathbf{x},t) \\ \mathbf{0} \end{pmatrix}, \tag{1.23}$$

if the vector of parameters is constant. The initial conditions are then

$$\mathbf{z}_\mathrm{e}(t_0) = \begin{pmatrix} \mathbf{z}_0(\mathbf{x}) \\ \mathbf{x} \end{pmatrix}, \text{ with } \mathbf{x} \in [\mathbf{x}]. \tag{1.24}$$

When only the initial conditions are undetermined, but known to belong to some box, guaranteed ODE solvers such as `COSY IV`, `VSPODE`, or `ValEncIA-IVP` perform quite well, since they are evaluating a Taylor development of the solution with interval remainder, this development being made also with respect to the initial condition.

Alternatively, one may enclose the solutions of (1.22) with uncertain $\mathbf{x} \in [\mathbf{x}]$ between a coupled pair of ODEs with deterministic initial conditions using Müller's theorem [19], see also Chapter **??** in this book. Any guaranteed tool for solving ODEs may then be used to solve this system.

1.4 Example

To illustrate the efficiency of the optimization techniques presented previously, we consider the estimation of the parameters of compartmental models. These models are widely used, *e.g.*, in biology to study metabolisms [7].

A compartmental model consists of a finite set of homogeneous subsystems, called compartments, which may exchange material between them and with the outside world. The evolution of the quantity of material in the compartments is described by a set of first-order ordinary differential equations, corresponding to conservation equations, usually assumed to be linear and time-invariant. These equations can be written in the form of a state equation.

Consider for example the model described by Figure 1.1. If $\mathbf{z} = (z_1, z_2)^\mathrm{T}$ is the vector of the quantities of material in the two compartments, its evolution is de-

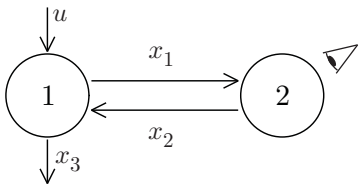

Fig. 1.1 Two-compartment model

scribed by the linear state equation

$$\begin{cases} \dot{z}_1 = -(x_1 + x_3) z_1 + x_2 z_2, \\ \dot{z}_2 = x_1 z_1 - x_2 z_2. \end{cases} \tag{1.25}$$

Assume that the initial state is known, and such that $\mathbf{z}_0 = (1,0)^{\mathrm{T}}$, that there exists some *true* parameter value $\mathbf{x}^*$, and that only Compartment 2 is observed so that the observation equation is

$$y(t_i) = y_{\mathrm{m}}(\mathbf{x}^*, t) + b(t_i), \quad i = 1, ..., N$$

with

$$y_{\mathrm{m}}(\mathbf{x}, t) = z_2(\mathbf{x}, t_i)$$

and the $b(t_i)$'s are some noise realizations.

To generate artificial data, a two-compartment model with $\mathbf{x}^* = (0.6, 0.15, 0.35)^{\mathrm{T}}$ has been simulated. The data were then obtained by rounding the value of $z_2(t_i)$ to the nearest two-digit number for $t_i = i\Delta t$, with $\Delta t = 1$ s and $i = 1, \ldots, 15$. The initial search domain is $[\mathbf{x}]_0 = [0.01, 1]^{\times 3}$.

Parameter estimation is performed by minimizing the cost function (1.6).

1.4.1 Using an explicit expression for the model output

1.4.1.1 Original formulation of the problem

For the two-compartment model of Figure 1.1, one may show that the model output satisfies

$$y_{\mathrm{m}}(\mathbf{x}, t_i) = \frac{x_1}{\sqrt{a(\mathbf{x})}} (e^{\lambda_1(\mathbf{x})t} - e^{\lambda_2(\mathbf{x})t}), \tag{1.26}$$

where

$$a(\mathbf{x}) = (x_3 - x_2 + x_1)^2 + 4x_1 x_2, \tag{1.27}$$

$$\lambda_1(\mathbf{x}) = -(x_3 + x_2 + x_1 - \sqrt{a(\mathbf{x})})/2, \tag{1.28}$$

$$\lambda_2(\mathbf{x}) = -(x_3 + x_2 + x_1 + \sqrt{a(\mathbf{x})})/2. \tag{1.29}$$

1.4.1.2 Reformulation of the problem

We rearrange $y_{\mathrm{m}}(\mathbf{x},t)$ as a product of univariate functions by factoring out $y_{\mathrm{m}}(\mathbf{x},t)$ as

$$y_{\mathrm{m}}(\mathbf{x},t) = \frac{x_1}{\sqrt{a(\mathbf{x})}} \left(e^{\sqrt{a(\mathbf{x})}t/2} - e^{-\sqrt{a(\mathbf{x})}t/2} \right) e^{-(x_3+x_2+x_1)t/2} \tag{1.30}$$

$$= (x_1 e^{-x_1 t/2}) \, (e^{-x_2 t/2}) \, (e^{-x_3 t/2}) \left(\frac{e^{\sqrt{a(\mathbf{x})}t/2} - e^{-\sqrt{a(\mathbf{x})}t/2}}{\sqrt{a(\mathbf{x})}} \right). \tag{1.31}$$

The exact range of $a(\mathbf{x})$ and each parenthesized term in (1.31) can be evaluated easily with interval arithmetic (except, of course, for outward rounding) over any box in search space. Therefore, overestimation is greatly reduced when compared to the original formulation.

In the COCONUT environment we introduced the following univariate functions, on which optimal enclosures can be computed for all derivatives and the inverse function needed for the constraint propagation (see Section 1.3.1).

$$\mathrm{xexp}(v,c) := v e^{cv}, \tag{1.32}$$

$$\mathrm{hsf}(v,c) := \frac{e^{-c\sqrt{v}} - e^{c\sqrt{v}}}{\sqrt{v}} = -\frac{2}{\sqrt{v}} \sinh(c\sqrt{v}). \tag{1.33}$$

In addition we added the special quadratic node

$$\mathrm{asqr}(\mathbf{v}) := (v_3 - v_2 + v_1)^2 + 4 v_1 v_2. \tag{1.34}$$

with exact range evaluations up to second order and inverse function evaluations. This can be generalized for arbitrary quadratic functions, see [3, 4].

With these new expressions and the exponential node (denoted here by $\exp(v,c) = e^{cv}$), $y_{\mathrm{m}}(\mathbf{x},t)$ is represented as

$$y_{\mathrm{m}}(\mathbf{x},t) = \mathrm{xexp}(x_1,c) \cdot \exp(x_2,c) \cdot \exp(x_3,c) \cdot \mathrm{hsf}(\mathrm{asqr}(\mathbf{x}),c), \tag{1.35}$$

with $c = -t/2$.

1.4.1.3 Symmetry breaking

Note that the cost function has a x_2–x_3 permutation symmetry; this follows easily from the x_2–x_3 symmetry of $a(\mathbf{x})$ and (1.31). This symmetry was already identified in [37]. The symmetry can be broken easily to reduce the necessary computations. For this purpose we implemented a new box elimination/pruning tool in `coco_gop_ex` just for the present problem instance: for each box $[\mathbf{x}]$ under processing we eliminate subregions of it for which $[x_2] > [x_3]$.

Here, the symmetry, which translates in a lack of structural identifiability of the model is detected easily before performing the optimization. The automatic identification and treatment of symmetries in models represented by DAGs is an ongoing research topic in the development of the COCONUT environment.

However, if symmetries or lack of structural identifiability is not detected *a priori*, an *a posteriori* detection is possible, by looking at the set of boxes containing the candidate global optimizers, which may consists of several disconnected components. This is a definite advantage of these identification approaches based on deterministic global optimization compared to local search techniques.

1.4.1.4 Results

In the present study we used `coco_gop_ex` [17], the bound-constrained interval B&B solver of the COCONUT environment [1, 27]. The COCONUT environment is a software platform for global optimization that provides various state-of-the-art modules that can be combined in strategies for solving global optimization problems. For bound-constrained problems the solver `coco_gop_ex` is provided which implements the B&B algorithm loosely described in Section 1.2.3.

For computing centered forms, *etc.*, COCONUT provides various algorithmic differentiation tools [28]. We computed $\nabla f([\mathbf{x}])$ by backward algorithmic differentiation. Second-order derivatives are computed through Hessian times vector products also with a backward evaluation scheme. The third-order derivatives are computed as follows: during the problem initialization, the DAG of the Karush-John first order necessary conditions to the problem is created; this DAG contains ∇f as a subgraph. To get third-order derivatives, the Hessian times vector product evaluator is applied on this subgraph. Alternatively, there is now a new third derivative evaluator which does not need the Karush-John conditions, but for the current test this has not yet been used.

For local optimization, interfaces to many different local solvers are provided by the COCONUT environment. For bound constrained problems, we use `LBFGS-B` [2].

For a detailed description of `coco_gop_ex` and the ways of synchronizing the tools used we refer to [17].

We solved the example problem on a PC with an Intel Dual-Core Mobile CPU at 1.73 GHz and with 2 GB RAM under Linux. To show how `coco_gop_ex` tackled the problem, in Table 1.1 we introduce results for different tolerance values ε from 10^{-2} to 10^{-9}. In each row of the table we gave the running time in seconds, the number of B&B iterations, the number of boxes in the result list, and the componentwise width of the search space for which we proved that it contains all global minimizers of the problem. (The latter information was computed as the componentwise hull of the elements of R^i.)

Our conclusions are the following: for larger tolerance values ($\varepsilon \geq 10^{-3}$) the problem was solved by mostly using pure splitting and first-order information. The result obtained for $\varepsilon = 10^{-3}$ with the new algorithm is similar to that of the basic

Table 1.1 Solutions of the two-compartment model problem with different tolerance values ε. In each row, CPU is the running time in seconds, NIT is the number of B&B iterations, NBoxes is the number of small result boxes, and w is the componentwise width of the hull of the enclosures of all global minimizers (rounded to 3 nonzero decimals).

ε	CPU	NIT	NBoxes	w
10^{-2}	14	1023	1191	$[7.65, 13.2, 22.2] \cdot 10^{-2}$
10^{-3}	71	5680	5581	$[5.62, 5.89, 12.8] \cdot 10^{-2}$
10^{-4}	176	14332	2625	$[3.28, 2.20, 5.89] \cdot 10^{-2}$
10^{-5}	218	16973	1602	$[7.54, 5.62, 15.5] \cdot 10^{-3}$
10^{-6}	251	18984	790	$[1.50, 1.22, 3.12] \cdot 10^{-3}$
10^{-7}	278	21024	724	$[4.06, 2.52, 6.34] \cdot 10^{-4}$
10^{-8}	283	21377	8	$[7.79, 5.48, 13.4] \cdot 10^{-5}$
10^{-9}	284	21377	0	$[3.25, 2.21, 6.40] \cdot 10^{-12}$

interval B&B algorithm used in [37], for which the solution was obtained in about 3 hours (on a slightly slower computer). Nevertheless, `coco_gop_ex` provides the solution in just over a minute, with an approximate speedup of more than 100 times. This is due to the symbolic transformations applied to the problem (*i.e.*, the special handling of the univariate subexpressions) and the efficiency of constraint propagation. Note that a very good approximation to the global minimum is found after only a few hundred iterations. Therefore the bound test and the CP module can work efficiently with it right from the beginning of the algorithm.

Second-order tools start to work efficiently for $\varepsilon \leq 10^{-4}$, when the processed boxes reach the size of what is approximately the size of the output boxes for 10^{-3}. For instance, we found that when solving with $\varepsilon = 10^{-3}$, only around 5% of the total amount of eliminated boxes were thrown away by the suboptimality test using second-order centered form updates, but for $\varepsilon = 10^{-4}$ this ratio was about 40%. Indeed, without using second-order information the cluster effect would have already dominated the search even for this tolerance value: when we disabled all second-order tools, we obtained the solution in around 12 minutes instead of 3, with over 64 000 result boxes! With second-order information the cluster effect is clearly avoided, as shown also by the drop in the number of output boxes from 5 600 (with $\varepsilon = 10^{-3}$) to about 2 600.

From this point the refinement of the solution with smaller and smaller tolerance values was relatively easy, *e.g.*, solving the problem with $\varepsilon = 10^{-9}$ instead of 10^{-4} took only about 110 more seconds, with continuous drops in both the number and the size of the result boxes. For $\varepsilon = 10^{-8}$ and $\varepsilon = 10^{-9}$ the boxes became small enough so that the exclusion box utility also took effect. The number 0 in the NIT column of the last row actually indicates that we have no boxes left outside the exclusion box (the componentwise widths in column w are thus the widths of the inclusion box belonging to that exclusion box). That is, at this point the solution became fully specified up to the maximal possible capabilities of our algorithm. As a summary, we found that the two-compartment example model has

– *one unique global minimizer* (apart from the $x_2 - x_3$ permutation symmetry), located in the interior of the (inclusion) box

$$\begin{pmatrix} [0.604961728242, 0.604961728246], \\ [0.144474180373, 0.144474180376], \\ [0.366021184203, 0.366021184210] \end{pmatrix}$$

(the result intervals are given outward rounded with 12 decimal digits), and
– the enclosure of the global minimum is

$$[6.72177710824, 6.72177710827] \cdot 10^{-5}.$$

The fact that the width of the enclosure of the global minimum is of the order of the machine epsilon shows that the algorithm has reached the maximal possible resolution with standard double-precision floating-point computation.

For solving this example problem, we used two tools that may not be applicable in general, namely, analytic reformulation to reduce the interval overestimation and symmetry breaking. We also solved the example problem without these two acceleration tools. Our final conclusions were precisely same as above, *i.e.*, with tolerance $\varepsilon = 10^{-9}$ we reached the tolerance of the unique optimal solution and the global optimum presented above. The performance indicators were, of course, different from those of the first run: the running time and the number or required iterations were around 6 and 4 times larger, respectively, while maximal number of result boxes (also peaking at $\varepsilon = 10^{-3}$) was around 4 times larger than in the fully accelerated method.

1.4.2 Without using an explicit expression for the model output

1.4.2.1 Sensitivity functions

All first-order sensitivity functions are easily derived from (1.25). These sensitivity functions are denoted $s_{ij}(\mathbf{x},t) = \partial z_i / \partial x_j$ in what follows. Sensitivity functions may be obtained by pairs, for example

$$\begin{cases} \frac{\mathrm{d}s_{11}}{\mathrm{d}t} = -z_1 - (x_1 + x_3)\, s_{11} + x_2 s_{21} \\ \frac{\mathrm{d}s_{21}}{\mathrm{d}t} = z_1 + x_1 s_{11} - x_2 s_{21} \end{cases} \tag{1.36}$$

with $s_{11}(0) = s_{21}(0) = 0$. However, (1.36) cannot be solved alone, as it requires to be coupled with (1.25). Thus, all first-order sensitivity function together with the system output require the solution of three coupled systems of ODEs of dimension 4.

1.4.2.2 Müller's theorem

When outer-bounding the range of the gradient of the cost over some box $[\mathbf{x}]$, one has to evaluate the set of values taken by the sensitivity functions over $[\mathbf{x}]$. For that purpose, Müller's theorem is used to get for each coupled system of four *uncertain* ODEs defined in Section 1.4.2.1, a coupled system of eight *deterministic* ODEs. For example, to get enclosures of the state and of the sensitivity function of the state with respect to x_1, one has to solve the following system of ODEs

$$\begin{cases} \frac{\mathrm{d}\underline{z}_1}{\mathrm{d}t} = -\left(\bar{x}_1+\bar{x}_3\right)\underline{z}_1+\underline{x}_2\underline{z}_2 \\ \frac{\mathrm{d}\underline{z}_2}{\mathrm{d}t} = \underline{x}_1\underline{z}_1-\bar{x}_2\underline{z}_2 \\ \frac{\mathrm{d}\bar{z}_1}{\mathrm{d}t} = -\left(\underline{x}_1+\underline{x}_3\right)\bar{z}_1+\bar{x}_2\bar{z}_2 \\ \frac{\mathrm{d}\bar{z}_2}{\mathrm{d}t} = \bar{x}_1\bar{z}_1-\underline{x}_2\bar{z}_2 \\ \frac{\mathrm{d}\underline{s}_{11}}{\mathrm{d}t} = -\bar{z}_1-\left(\bar{x}_1+\bar{x}_3\right)\underline{s}_{11}+\underline{x}_2\underline{s}_{21} \\ \frac{\mathrm{d}\underline{s}_{21}}{\mathrm{d}t} = \underline{z}_1+\underline{x}_1\underline{s}_{11}-\bar{x}_2\underline{s}_{21} \\ \frac{\mathrm{d}\bar{s}_{11}}{\mathrm{d}t} = -\underline{z}_1-\left(\underline{x}_1+\underline{x}_3\right)\bar{s}_{11}+\bar{x}_2\bar{s}_{21} \\ \frac{\mathrm{d}\bar{s}_{21}}{\mathrm{d}t} = \bar{z}_1+\bar{x}_1\bar{s}_{11}-\underline{x}_2\bar{s}_{21} \end{cases} \tag{1.37}$$

with $\underline{z}_1(0)=\bar{z}_1(0)=1$, $\underline{z}_2(0)=\bar{z}_2(0)=0$, and $\underline{s}_{11}(0)=\underline{s}_{21}(0)=\bar{s}_{11}(0)=\bar{s}_{21}(0)=0$. At any time instant t_i, when $\mathbf{x}\in[\mathbf{x}]$, an enclosure for $z_1([\mathbf{x}],t_i)$ is given by $[\underline{z}_1([\mathbf{x}],t_i),\bar{z}_1([\mathbf{x}],t_i)]$, obtained by solving (1.37). Similar enclosures are obtained for $z_2([\mathbf{x}],t_i)$, $s_{12}([\mathbf{x}],t_i)$, $s_{22}([\mathbf{x}],t_i)$, $s_{13}([\mathbf{x}],t_i)$, and $s_{23}([\mathbf{x}],t_i)$.

1.4.2.3 Results

Only first-order sensitivity functions have been used. Guaranteed numerical integration of systems such as (1.37) has been performed using VNODE-LP. Thus, only basic box elimination tests using first-order derivatives were implemented. An equivalent of the PAID contractor has been employed using centered forms for the cost function.

For a tolerance parameter $\varepsilon=0.001$, a list of boxes is obtained whose projections onto the (x_1,x_2) plane and (x_2,x_3) plane are shown in Figure 1.2. Only boxes for which it was not possible to prove that they do not contain any global minimizer are represented, so this is an outer approximation. This result has been obtained in 3 h on a Pentium IV at 2 GHz. The set of boxes contains the solution provided in Section 1.4.1.4. The cluster effect could not be avoided here due to the lack of use of higher-order methods.

The time required to obtain the solution is much higher (more than 100 times for $\varepsilon=0.001$) than that required in Section 1.4.1.4 due to the necessity to perform numerical integration.

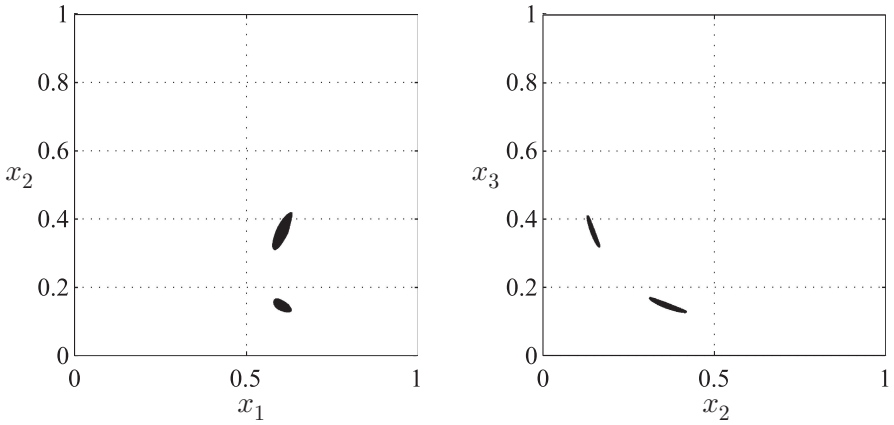

Fig. 1.2 Projection onto the (x_1,x_2) plane (left) and (x_2,x_3) plane (right) of a guaranteed outer approximation of the set of all global minimizers ($\varepsilon = 0.001$)

1.5 Conclusions and perspectives

Interval analysis provides tools for the guaranteed characterization of the set of all global minimizers of the cost function associated with parameter estimation even when the model output is obtained via the numerical solution of a set of ordinary differential equations. This chapter has shown that when the cost function involves many occurrences of the parameters, as is usually the case in parameter estimation via optimization, higher-order techniques for box reduction and elimination, as well as reformulation of the cost function may play a very important role in reducing complexity.

Such tools are however still very difficult to employ when no explicit expression of the cost function is available. Their adaptation to models described by ODEs poses in principle no problem. It would for example be possible to use higher-order Taylor models [16]. Such models would be helpful to get closed-form enclosures of the cost function. Higher-order Taylor models have also better approximation properties than the methods described here. However, computing times are $O(n^k)$ in dimension n for order k. This usually takes too much time already in low dimensions, except if these models are used near the solution only. There, however, we use the exclusion box trick of Section 1.3.2 which is usually sufficient and can be performed with $O(n^3)$ effort. The combination of higher-order Taylor models with sensitivity functions, to get closed-form enclosures for gradients and Hessian matrices has also to be considered.

Acknowledgements This work was supported by the Austrian Science Found (FWF) Grants Nr. P18704-N13 and P22239-N13, by the Hungarian National Development Agency (NFÜ) Grant TÁMOP-4.2.2/08/1/2008-0008, by the ANR CPP, and by the Digiteo PASO project.

References

1. The COCONUT environment. URL www.mat.univie.ac.at/coconut-environment
2. Byrd, R., Lu, P., Nocedal, J., Zhu, C.: A limited memory algorithm for bound constrained optimization. SIAM J. Sci. Comput. **16**, 1190–1208 (1995)
3. Domes, F., Neumaier, A.: Rigorous enclosures of ellipsoids and directed cholesky factorizations (2009). URL http://www.mat.univie.ac.at/ dferi/publ/Cholesky.pdf. Manuscript
4. Domes, F., Neumaier, A.: Constraint propagation on quadratic constraints. Constraints **10**, 404–429 (2010)
5. Hansen, E.R.: Global Optimization Using Interval Analysis. Marcel Dekker, New York, NY (1992)
6. Hoefkens, J., Berz, M., Makino, K.: Efficient high-order methods for ODEs and DAEs. In: G. Corliss, C. Faure, A. Griewank (eds.) Automatic Differentiation : From Simulation to Optimization, pp. 341–351. Springer-Verlag, New-York, NY (2001)
7. Jacquez, J.A.: Compartmental Analysis in Biology and Medicine. Elsevier, Amsterdam (1972)
8. Kahan, W.: A more complete interval arithmetic. Lecture notes for a summer course, University of Toronto, Canada (1968)
9. Kay, S.M.: Fundamentals of Statistical Signal Processing, Volume I: Estimation Theory. Prentice Hall (1993)
10. Kearfott, R.B.: INTLIB: a portable FORTRAN 77 elementary function library. ACM Transactions on Mathematical Software **20**(4), 447–459 (1994)
11. Kearfott, R.B.: Rigorous Global Search: Continuous Problems (Nonconvex Optimization and Its Applications. Springer-Verlag (2010)
12. Land, A.H., Doig, A.G.: An automated method for solving discrete programming problems. Econometrica **28**, 497–520 (1960)
13. Lin, Y., Stadtherr, M.A.: Validated solution of ODEs with parametric uncertainties. In: W. Marquardt, C. Pantelides (eds.) Proc. 16th European Symposium on Computer Aided Process Engineering and 9th International Symposium on Process Systems Engineering, *Computer Aided Chemical Engineering*, vol. 21, pp. 167 – 172. Elsevier (2006). DOI 10.1016/S1570-7946(06)80041-6. URL http://www.sciencedirect.com/science/article/B8G5G-4P37F2K-S/2/a22680956d2786784b23e88e9b272db6
14. Little, J.D., Murty, K.C., Sweeney, D.W., Karel, C.: An algorithm for the travelling salesman problem. Operations Research **11**, 972–989 (1963)
15. Lohner, R.: Computation of guaranteed enclosures for the solutions of ordinary initial and boundary value-problem. In: J.R. Cash, I. Gladwell (eds.) Computational Ordinary Differential Equations, pp. 425–435. Clarendon Press, Oxford (1992)
16. Makino, K., Berz, M.: Taylor models and other validated functional inclusion methods. International Journal **4**(4), 379–456 (2003)
17. Markót, M.C., Schichl, H.: Bound constrained optimization in the COCONUT environment (2010). Manuscript
18. Moore, R.E.: Interval arithmetic and automatic error analysis in digital computing. Phd thesis, Stanford University, Stanford, CA (1962). URL http://interval.louisiana.edu/Moores early papers/disert.pdf
19. Müller, M.: Über das Fundamentaltheorem in der Theorie der gewöhnlichen Differentialgleichungen. Math. Z. **26**, 619–645 (1926)
20. Nedialkov, N.S., Jackson, K.R.: Methods for initial value problems for ordinary differential equations. In: U. Kulisch, R. Lohner, A. Facius (eds.) Perspectives on Enclosure Methods, pp. 219–264. Springer-Verlag, Vienna (2001)
21. Neumaier, A.: Interval Methods for Systems of Equations. Cambridge University Press, Cambridge, UK (1990)
22. Neumaier, A.: Complete search in continuous global optimization and constraint satisfaction. In: A. Iserles (ed.) Acta Numerica, pp. 271–369. Cambridge University Press (2004)

23. Ratschek, H., Rokne, J.: New Computer Methods for Global Optimization. Ellis Horwood, Chichester, UK (1988)
24. Ratz, D., Csendes, T.: On the selection of subdivision directions in interval branch-and-bound methods for global optimization. Journal of Global Optimization **7**(2), 183–207 (1995)
25. Rauh, A., Auer, E., Minisini, J., Hofer, E.P.: Extensions of ValEncIA-IVP for reduction of overestimation, for simulation of differential algebraic systems, and for dynamical optimization. Proc. Appl. Math. Mech. **7**(1), 1023,0011023,002 (2007). DOI 10.1002/pamm.200700022
26. Rauh, A., Hofer, E.P., Auer, E.: VALENCIA-IVP: A comparison with other initial value problem solvers. In: Proc. 12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics (SCAN 2006), p. 36. IEEE Computer Society, Duisburg, Germany (2006). DOI 10.1109/SCAN.2006.47
27. Schichl, H.: Global optimization in the COCONUT project. In: Proceedings of the Dagstuhl Seminar "Numerical Software with Result Verification", *Lecture Notes in Comput. Sci.*, vol. 2991, pp. 277–293 (2004)
28. Schichl, H., Markót, M.C.: Algorithmic differentiation in the COCONUT environment (2010). URL http://www.mat.univie.ac.at/ herman/papers/griewank.pdf. Manuscript
29. Schichl, H., Markót, M.C.: Interval analysis on directed acyclic graphs for global optimization. higher order methods (2010). URL http://www.mat.univie.ac.at/ herman/papers/dag2.pdf. Manuscript
30. Schichl, H., Markót, M.C.: Optimal enclosures of derivatives and slopes for univariate functions (2010). URL http://www.mat.univie.ac.at/ herman/papers/univar.pdf. Manuscript
31. Schichl, H., Markót, M.C., Neumaier, A.: Exclusion regions for optimization problems (2010). URL http://www.mat.univie.ac.at/ herman/papers/exclopt.pdf. Manuscript
32. Schichl, H., Neumaier, A.: Exclusion regions for systems of equations. SIAM J. Numer. Anal. **42**, 383–408 (2004)
33. Schichl, H., Neumaier, A.: Interval analysis on directed acyclic graphs for global optimization. J. Global Optim. **33**, 541–562 (2006)
34. Shen, Z., Neumaier, A.: The Krawczyk operator and Kantorovich's theorem. J. Math. Anal. Appl. **149**, 437–443 (1990)
35. Skelboe, S.: Computation of rational interval functions. BIT **14**, 87–95 (1974)
36. Vu, X.H., Schichl, H., Sam-Haroud, D.: Using directed acyclic graphs to coordinate propagation and search for numerical constraint satisfaction problems. In: Proc. 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004). Florida (2004)
37. Walter, E., Kieffer, M.: Guaranteed optimisation of the parameters of continuous-time knowledge-based models. In: C. Commault, N. Marchand (eds.) Positive Systems, *LNCIS*, vol. 341, pp. 137–144. Springer, Heidelberg (2006)
38. Walter, E., Pronzato, L.: Identification of Parametric Models from Experimental Data. Springer-Verlag, London (1997)
39. Wolfe, M.A.: Interval methods for global optimization. Applied Mathematics and Computation **75**, 179–206 (1996)